









































*# Задание 1  
import math  
  
  
def main(d):  
 y = float(d)  
 z = math.pow(math.log(y, 10), 4)  
 m = (math.pow((79 \* y - 51 \* math.pow(y, 3)), 5) + math.atan(y) / 75)  
 f = (24 \* math.pow((28 \* y - (y \* y / 39)), 5) + math.pow(y, 4))  
 g = (8 \* math.pow(y, 3))  
 k = math.sqrt(f / g)  
 return z / m + k*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*# Задание 2  
import math  
  
  
def main(d):  
 y = float(d)  
 if y < 6:  
 z = 1-72\*(math.pow(28\*y+math.pow(y, 3), 7))  
 elif 6 <= y < 23:  
 z = 97\*math.pow((y\*y+13+(y/52)), 4)+47\*math.exp(5)\*y\*y  
 else:  
 z = 47\*math.pow(math.atan(y/71), 6) - 1  
 return z

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*# Задание 3  
import math

def main(a, m):  
 a = int(a)  
 m = int(m)  
 s1 = 0  
 s2 = 0  
 for c in range(1, a+1):  
 s1 += (1+73\*math.pow(math.sin(c), 4))  
 for k in range(1, m+1):  
 for i in range(1, a+1):  
 s2 += math.pow(math.log(k, 10), 4)  
 s2 += math.pow((k\*k-40\*math.pow(i, 3)-1), 5)  
 return s1 - s2  
  
print(main(8, 5))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Задание 4  
import math  
  
  
def main(d):  
 n = int(d)  
 if n == 0:  
 return 0.01  
 else:  
 k = math.atan(math.pow(main(n-1), 3)-1)  
 return 1 + 45\*math.pow(k, 3) + main(n-1)  
  
print(main(6))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*# Задание 5  
import math  
  
  
def main(x):  
 n = len(x)  
 s1 = 0  
 for i in range(1, n+1):  
 k = math.pow(x[n-i], 3)  
 m = 50\*math.pow(x[n-i], 2)  
 s1 += 4\*math.pow((k - m), 5)  
 return s1  
  
print(main([-0.37, 0.42]))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*# Задание 6  
def main(arr):  
 m = [  
 [1999, 'BOO', 1969, 0],  
 [1999, 'BOO', 1986, 1],  
 [1999, 'BOO', 2000, 2],  
 [1999, 'NIX', 1992, 3],  
 [1999, 'NIX', 1987, 4],  
 [1999, 'NIX', 1986, 5],  
 [1985, 'BOO', 1969, 6],  
 [1985, 'BOO', 1986, 7],  
 [1985, 'BOO', 2000, 8],  
 [1985, 'NIX', 1992, 9],  
 [1985, 'NIX', 1987, 10],  
 [1985, 'NIX', 1986, 11],  
 ]  
 k = 0  
 for i in range(12):  
 if arr[0] == m[i][0]:  
 if arr[1] == m[i][1] and arr[1] == 'BOO':  
 if arr[2] == m[i][2]:  
 return m[i][3]  
 elif arr[1] == m[i][1] and arr[1] == 'NIX':  
 if arr[3] == m[i][2]:  
 return m[i][3]  
  
  
print(main([1985, 'NIX', 1986, 1986]))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Задание 7

def main(bit\_fields):  
 result = 0  
 result |= int(bit\_fields[0][1])  
 result |= int(bit\_fields[1][1]) << 10  
 result |= int(bit\_fields[2][1]) << 12  
 return result  
  
print(main([('H1', '233'), ('H3', '0'), ('H4', '109')]))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Задание 8

def main(x):  
 x = x.replace('begin', '')\  
 .replace('<<', '')\  
 .replace('local', '')\  
 .replace('>>', '')\  
 .replace('\'', '')\  
 .replace('\n', '')\  
 .replace(' ', '')\  
 .replace('#', '')  
 x\_parts = x.split('end')  
 x\_parts.pop(-1)  
 x\_parts1 = [i.split('::=') for i in x\_parts]  
 result = []  
 for i in range(len(x\_parts1)):  
 value = int(x\_parts1[i][1])  
 key = x\_parts1[i][0]  
 result.append((key, value))  
 return result

# Testing  
print(main('<<begin local inbia ::= #-7934 end begin local xedibe\_95 ::=#5519 end\nbegin local bicein::=#-9011 end begin local ceedes ::= #-2730 end >>'))

# Задание 9

def main(args):  
 result = []  
 for i in args:  
 row = []  
 if i[1] == '1':  
 row.append('true')  
 else:  
 row.append('false')  
 row.append(i[2].replace('-', '/'))  
 row.append(i[3].split('@')[1])  
 if row not in result:  
 result.append(row)  
 return result  
  
  
# Testing  
print(main([[None, '1', '24-02-03', 'rodusic5@mail.ru'], [None, '0', '05-09-00', 'facikak6@yahoo.com'], [None, '1', '24-02-03', 'rodusic5@mail.ru']]))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Задание 10

class MealyError(Exception):  
 def \_\_init\_\_(self, method\_name):  
 self.method\_name = method\_name  
  
  
class StateMachine:  
 state = 'A'  
  
 def reset(self):  
 if self.state == 'A':  
 self.state = 'B'  
 return 0  
 elif self.state == 'B':  
 self.state = 'C'  
 return 2  
 elif self.state == 'D':  
 self.state = 'B'  
 return 6  
 elif self.state == 'F':  
 self.state = 'D'  
 return 8  
 else:  
 raise(MealyError('reset'))  
  
 def tread(self):  
 if self.state == 'B':  
 return 3  
 elif self.state == 'A':  
 self.state = 'D'  
 return 1  
 elif self.state == 'C':  
 self.state = 'D'  
 return 4  
 elif self.state == 'D':  
 self.state = 'E'  
 return 5  
 else:  
 raise(MealyError('tread'))  
  
 def roam(self):  
 if self.state == 'E':  
 self.state = 'F'  
 return 7  
 else:  
 raise(MealyError('roam'))  
  
  
def main():  
 return StateMachine()  
  
  
def test():  
 sm = main()  
 # exception test  
 sm.state = 'A'  
 try:  
 sm.roam()  
 except MealyError:  
 pass  
 sm.state = 'B'  
 try:  
 sm.roam()  
 except MealyError:  
 pass  
 sm.state = 'C'  
 try:  
 sm.roam()  
 except MealyError:  
 pass  
 sm.state = 'C'  
 try:  
 sm.reset()  
 except MealyError:  
 pass  
 sm.state = 'D'  
 try:  
 sm.roam()  
 except MealyError:  
 pass  
 sm.state = 'E'  
 try:  
 sm.reset()  
 except MealyError:  
 pass  
 sm.state = 'E'  
 try:  
 sm.tread()  
 except MealyError:  
 pass  
 sm.state = 'F'  
 try:  
 sm.roam()  
 except MealyError:  
 pass  
 sm.state = 'F'  
 try:  
 sm.tread()  
 except MealyError:  
 pass  
 sm.state = 'A'  
 assert sm.reset() == 0  
 assert sm.state == 'B'  
 sm.state = 'A'  
 assert sm.tread() == 1  
 assert sm.state == 'D'  
 sm.state = 'B'  
 assert sm.reset() == 2  
 assert sm.state == 'C'  
 sm.state = 'B'  
 assert sm.tread() == 3  
 assert sm.state == 'B'  
 sm.state = 'C'  
 assert sm.tread() == 4  
 assert sm.state == 'D'  
 sm.state = 'D'  
 assert sm.reset() == 6  
 assert sm.state == 'B'  
 sm.state = 'D'  
 assert sm.tread() == 5  
 assert sm.state == 'E'  
 sm.state = 'E'  
 assert sm.roam() == 7  
 assert sm.state == 'F'  
 sm.state = 'F'  
 assert sm.reset() == 8  
 assert sm.state == 'D'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Задание 11

import struct  
import pprint  
  
  
def parse\_e(data, pointer):  
 e1 = struct.unpack('>Q', data[pointer:pointer + 8])[0]  
 e2 = list(struct.unpack('>3b', data[pointer + 8:pointer + 11]))  
 e3 = struct.unpack('>B', data[pointer + 11:pointer + 12])[0]  
 return {'E1': e1, 'E2': e2, 'E3': e3}  
  
  
def parse\_d(data, pointer):  
 d1, d2, d3, d4 = list(struct.unpack('>fIih', data[pointer:pointer + 14]))  
 d5, d6, d7, d8 = list(struct.unpack('>fhQd',  
 data[pointer + 14:pointer + 36]))  
 return {'D1': d1, 'D2': d2, 'D3': d3, 'D4': d4,  
 'D5': d5, 'D6': d6, 'D7': d7, 'D8': d8}  
  
  
def parse\_c(data, pointer):  
 c1 = struct.unpack('>b', data[pointer:pointer + 1])[0]  
 c2 = list(struct.unpack('>2h', data[pointer + 1:pointer + 5]))  
 return {'C1': c1, 'C2': c2}  
  
  
def parse\_b(data, pointer):  
 b1, b2, b3, b4 = list(struct.unpack('>QbbB', data[pointer:pointer + 11]))  
 b5 = list()  
 f5 = struct.unpack('>7I', data[pointer + 11:pointer + 39])  
 for i in range(7):  
 b5.append(parse\_c(data, f5[i]))  
 return {'B1': b1, 'B2': b2, 'B3': b3, 'B4': b4,  
 'B5': b5}  
  
  
def parse\_a(data, pointer):  
 a1 = ''.join(map(str, struct.unpack('>8c',  
 data[pointer:pointer + 8])))  
 a1 = a1.replace('\'', '')[1::2]  
 f2 = struct.unpack('>I', data[pointer + 8:pointer + 12])[0]  
 a2 = parse\_b(data, f2)  
 f3 = struct.unpack('>H', data[pointer + 12:pointer + 14])[0]  
 a3 = parse\_d(data, f3)  
 f4 = struct.unpack('>I', data[pointer + 14:pointer + 18])[0]  
 a4 = parse\_e(data, f4)  
 a5, a6 = list(struct.unpack('>bQ', data[pointer + 18:pointer + 27]))  
 f7 = struct.unpack('>II', data[pointer + 27:pointer + 35])  
 a7 = list(struct.unpack(f'>{f7[0]}d', data[f7[1]:f7[1] + f7[0] \* 8]))  
 a8 = struct.unpack('>f', data[pointer + 35:pointer + 39])[0]  
 return {'A1': a1, 'A2': a2, 'A3': a3, 'A4': a4,  
 'A5': a5, 'A6': a6, 'A7': a7, 'A8': a8}  
  
  
def main(data):  
 return parse\_a(data, 4)  
  
  
# Testing  
pprint.pprint(main(b'MBUAjoarbobh\x00\x00\x00N\x00u\x00\x00\x00\x99\xdac\xbdV\x087\x1c\xcfw\x00'  
 b'\x00\x00\x02\x00\x00\x00\xa5\xbe[\xea[>\x13&\xd00\xeaeO\xd5n\xb3r\xb6\x12iHW'  
 b'\*\xb5\xb7\x9cke\xd9\xa0\xd9\xe4\xe4\x0bA\xc9\xd0v\x91\xa7\xb9\x90'  
 b's\x08\xc4\x8ac\xcd\xcf\xa3\xd4\x00\x00\x00+\x00\x00\x000\x00\x00\x00'  
 b'5\x00\x00\x00:\x00\x00\x00?\x00\x00\x00D\x00\x00\x00I\xbf$\xd2'  
 b'\xc8\x00\xce\x9a\x04h\x9f\xc2\xd9\x04\xbb\xbfjM\x0b\xe7A\x03\xb6\xcd'  
 b'\xb7\x87\xd7\xbcw?\xe1\x1a\xc6\xf7T6\xd4\xf5\x80\xe3\xea\x99\x90s)\xdbj\xf1'  
 b'\x93\xbf\xed#O\xcb\x8c!\xb8?\xef^\xb7\xc6\x91y`'))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def main(\*r):  
 s = ({'PIC', 'DART', 'COBOL', 'NUMPY'},  
 {'PIC', 'DART', 'COBOL', 'HY'},  
 {'PIC', 'DART', 'CUDA'},  
 {'PIC', 'DART', 'XS', 'NUMPY'},  
 {'PIC', 'DART', 'XS', 'HY'},  
 {'PIC', 'CSON'},  
 {'OCAML', 'DART'},  
 {'OCAML', 'CSON', 'NUMPY', 'COBOL'},  
 {'OCAML', 'CSON', 'NUMPY', 'CUDA'},  
 {'OCAML', 'CSON', 'NUMPY', 'XS'},  
 {'OCAML', 'CSON', 'HY'},  
 {'EC'})  
 s1 = set(\*r)  
 return [i for i in range(len(s)) if not (len(s[i] - s1))][0]  
  
  
print(main(['CSON', 'CUDA', 'NUMPY', 1994, 'OCAML']))  
print(main(['CSON', 'COBOL', 'NUMPY', 2000, 'EC']))  
print(main(['DART', 'CUDA', 'HY', 2000, 'PIC']))  
print(main(['CSON', 'XS', 'HY', 2000, 'PIC']))  
print(main(['CSON', 'COBOL', 'NUMPY', 2000, 'OCAML']))